APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: **DATA COMPRESSION SYSTEM AND TECHNIQUE**

APPLICANT: **CHRISTOPHER P. HONDL, JON D. CLAUSON**

DATA COMPRESSION SYSTEM AND TECHNIQUE

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims benefit of U.S. Provisional Application No. 60/211,493, filed June 14, 2000, which is incorporated herein by reference.

BACKGROUND

This invention relates to compression of data.

In dictionary-based compression techniques, a codebook or dictionary is generated during compression – the codebook assigns a unique code to a sequence of uncompressed data. Generally, the codebook must be stored along with the compressed data; otherwise a decompressor would have no way of knowing what the codes represent. In contrast, in LZW (Lempel-Ziv-Welsh) compression, the compressor and decompressor build identical codebooks as data is processed sequentially, thus avoiding the need to store or transmit a codebook. The compressor outputs a pattern code only after it has found the pattern more than once. The first time the compressor processes a sequence of data, it places that sequence in its codebook and outputs the sequence without any encoding. The decompressor will receive this sequence and place it in its codebook. The compressor, when it sees a pattern repeated for a second time, outputs the code from its codebook for the pattern. The decompressor can recognize the code because it has built an identical codebook from the previous sequences of data.

The LZW compression technique works well on a variety of data. The Unix *compress* utility and the personal computer *ARC* utility are based on LZW compression. Additionally, the Graphical Interchange Format (GIF), which is a popular palettized image compression format used in the World Wide Web (WWW), uses the LZW algorithm.

SUMMARY

An image is defined by pixels, where each pixel has a true color. The image is compressed such that the compressed image may be decompressed by a decompression method according to a color table-based compression technique. The image and a color table are received. The color table defines a mapping from true colors to index color values. A set of zero or more candidate strings for a current pixel in the image is identified in a

compression dictionary. Each candidate string corresponds to a string of pixels in the image, with the last pixel of the string corresponding to the current pixel. Each candidate string approximately matches the corresponding image pixel string. If the set of candidate strings for the current pixel is empty, one of the candidate strings for the previous current pixel is selected, and a code for the selected string is added to a compressed representation of the image.

Implementations may include one or more of the following features. The compressed representation may be embedded in a GIF file.

Two strings may be an approximate match if, at each string position, the true color from one of the strings approximately matches the true color from the other of the strings. Two colors may be approximate matches if a distance between them is less than a threshold. The distance between two colors may be measured using a standard Euclidean distance metric. The distance between two colors may be measured using a weighted Euclidean distance metric. The threshold may vary across the image.

Two strings may be approximate matches if an aggregate distance between the two strings is less than a threshold value.

The selected dictionary-based compression technique may implement an LZW algorithm. The candidate string that is selected may be a string whose selection causes a minimum increase in the size of an LZW dictionary.

An error amount may also be calculated. The error amount is calculated as the amount between the true color of the candidate string pixel corresponding to the current pixel and the true color of the current pixel. The error amount may be distributed to modify pixels in the image before processing them for compression. The error amount may be distributed according to an error diffusion technique. The error diffusion technique may be a Floyd-Steinberg technique.

Implementations of the invention may include one or more of the following advantageous features. For example, the compressed representation output from the compression process may be decompressed using standard LZW decompression, which is a very popular technique used in the WWW for GIF.

The details of one or more embodiments are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description, the drawings, and the claims.

DESCRIPTION OF DRAWINGS

Fig. 1 is a flow chart of a process for compressing an image using a dictionary-based technique in accordance with the invention.

Fig. 2 illustrates a color table used in the process of Fig. 1.

Fig. 3 illustrates a compression dictionary used in the process of Fig. 1.

Fig. 4 illustrates a sample compression dictionary used in the process of Fig. 1.

Fig. 5 is a diagram of a 3-pixel × 3-pixel image demonstrating base color estimations of the pixels.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

Fig. 1 shows a process 100 implemented in a computer program application for compressing data from an image. The process operates so as to generate a compressed representation that can be decompressed by a method according to a color table-based compression technique such as LZW.

In general, the process 100 operates by examining a compression dictionary for strings that the LZW compression algorithm uses for compression and then identifying approximate matches between the strings in that compression dictionary and strings of true colors of the image pixels. Optionally, a color of a particular pixel is adjusted during the process to account for approximation errors of pixels nearby the particular pixel.

Initially into the process 100, a digital image is received (step 102). A digital image is a collection of digital information that may be cast into the form of a visual image. Digital images may include, for example, photographs, artwork, documents, and web pages. Digital images may be obtained, for example, from digital cameras, digital video, scanners, and facsimile. The images may be two-dimensional or multi-dimensional. For example, three-dimensional images may include representations of three-dimensional space, or of two-dimensional movies, where the third dimension is time.

The fundamental element of a digital image is a pixel. Generally, a pixel has a specific location in the digital image and it contains color information for that location. Color information is represented by a vector of values, the vector characterizing all or a portion of the image intensity information. Color information could, for example, be

represented by red (R), green (G), and blue (B) intensities in an RGB color space. Or, color information may represent a single luminosity in a grayscale color space.

A color space is a multi-dimensional space in which each point in the space corresponds to a color. For example, RGB color space is a color space in which each point is a color formed of the additive amounts of red, green, and blue colors. As another example, color information could represent information such as cyan-magenta-yellow (CMY), cyan-magenta-yellow-black (CMYK), Pantone, Hexachrome, x-ray, infrared, and gamma ray intensities from various spectral wavelength bands. Thus, for example, in CMY color space, each point is a color formed of the combination of cyan, magenta, and yellow colors. Color information may, in addition, represent other modalities of information, such as acoustic amplitudes (sonar, ultrasound) or magnetic resonance imaging (MRI) amplitudes.

In RGB color space, levels of red, green, and blue can each range from 0 to 100 percent of full intensity. Each level can be represented by a range of decimal numbers from, for example, 0 to 255 (256 levels for each color) with an equivalent range of binary numbers extending from 00000000 to 11111111. The total number of available colors would therefore be 256 × 256 × 256, or 16,777,216 possible colors.

Because of the large number of available colors in color representations such as CMY, CMYK, LAB, XYZ, and RGB, and because of the varying degrees of resolution (for example, 12 bit or 16 bit depth), colors are often referred to as true colors. A true color is a direct representation of a color and is a point in the color space used to represent the color information of the image.

The process 100 operates by receiving a color table, also referred to as a color-lookup table (step 104). The color table generally defines a color palette that is used to represent an image as an indexed image. The color table is typically predefined, but may be built on the fly. Referring also to Fig. 2, the color table 200 includes entries, where each entry may be represented by index i 202 and a corresponding true color 204, which may be, for example, represented as a triplet for a three color space (x_i, y_i, z_i). In this case, each x_i, y_i, or z_i represents a color level in the three-color space. Because the number of true colors is generally large (as mentioned above, there are 16,777,216 true colors in a 256-cubed RGB color space), the color table 200 typically has entries for a subset of all available true colors, where the subset has N + 1 entries. N may be any value, but typical values are 255 (for 256 entries) or 4095 (for 4096 entries). In the web safe color palette, N is 215 (for 216 entries).

For GIF formats, N ranges from 1 to 255. Other dictionary formats may allow for larger ranges of N.

After receiving the color table, a compression dictionary is initially set up (step 106). The compression dictionary may be set up to include the color table (as shown in Fig. 3) or the compression dictionary may be set up to be empty. Referring also to Fig. 3, the compression dictionary 300 includes entries, where each entry is represented by a code j 302 and a corresponding string of true color codes $[(TC_1), (TC_2), \ldots]_j$ 304, where TC_k is a true color index selected from the subset N + 1 of all available true colors represented in the color table.

Referring to the sample compression dictionary 400 in Fig. 4, a TC_1 index of 72 may be, for example, the triplet associated with pale red in RGB color space (250, 75, 75) and a TC_2 index of 213 may be, for example, the triplet associated with the color dark green (64, 267, 84). In the sample compression dictionary 400, the code 456 indicates the string [72, 213], which corresponds to the index for pale red followed by the index for dark green.

The number of entries in the compression dictionary 300 at any time during compression is N + M + 1, where M is the total number of entries added during compression of the image up to that time. As discussed in greater detail below, the compression dictionary grows as new entries are added to compress the data.

The index 202 and codes 302 may be of variable length to further compress the data, as is well known in LZW compression. For example, the codes may start with the number of bits needed to represent the largest value plus 1 bit. If, while building the compression dictionary, a code added to the compression dictionary totally uses the current code length, then the code length may be increased by 1 bit. For example, if the initial compression dictionary is created to handle 256 entries (thus, the color table has 256 entries), then the code lengths would start with 9 bits. After the 511^{th} entry into the compression dictionary, the code lengths would automatically increment to 10 bits, and so on.

Before compression, a prefix string [prefix] is set to empty (step 108). The prefix string may be a sequential ordered list of indices from the color table. When referring to a string of data values present in the compression dictionary, that string of data values may be represented as [prefix][data value], indicating a concatenation of a prefix string with the data value string.

Next, a first pixel from the image is selected as the current pixel to consider (step 110). Although any pixel may be initially selected, it is conventional, when using GIF format, to select that pixel at an upper, left hand corner of the image, and to advance sequentially from left to right and top to bottom during the process 100.

The process 100 operates by identifying a base color of the current pixel (step 112). The base color may be identified using an error diffusion technique, which is based on the true color of the current pixel and a dither value of the current pixel, the dither value being a function of error values in the neighboring pixels. An error value of a pixel is a representation of how well a true color of that pixel matches a true color from the color table. Therefore, the dither value takes into account a total error in the color approximations of neighboring pixels. In other words, an error value at a pixel is distributed among other pixels in the image. One example of an error diffusion technique is the Floyd-Steinberg algorithm.

Referring to a 3-pixel × 3-pixel image 500 in Fig. 5, each pixel is represented by a square with a true color (TC), a base color (BC), and an error value. In Fig. 5, the kernel used for dithering includes only those pixels neighboring a particular pixel that are directly beside or above, and sequentially before the particular pixel. In general, the kernel used for dithering may be more complicated. For example, the kernel may include pixels neighboring a particular pixel that are diagonal from the particular pixel, or those pixels within a predetermined distance from the particular pixel.

In Fig. 5, arrows from one value to another represent dependency between those values. Each pixel's base color is dependent on that pixel's true color plus a fraction of the error values of neighboring pixels. Thus, the base color of pixel 5 depends on the its own true color and the percentages of each of the error values of pixels 2 and 4. In one error diffusion technique, the error value (EV) of a neighboring pixel is computed to be a percentage of the difference between the true color (TC) of that neighboring pixel input at step 102 and the true color approximately matching the base color of that neighboring pixel determined at a later step in the process 100.

Next, the process 100 identifies an index value of the current pixel (step 114). The index value of the current pixel is an index selected from the color table that corresponds to a true color that is nearest in color space to the base color of the current pixel. Any distance metric, such as a standard Euclidean metric or a weighted Euclidean metric, may be used to

measure the distance between the true color in the color table and the base color of the current pixel.

Next, a set of candidate strings is identified from the compression dictionary (step 116). Each candidate string is written as [candidate prefix][TC_C], where the index [TC_C] indicates a true color approximately matching the base color of the current pixel and [candidate prefix] is a prefix string. Identification of the set of candidate strings begins with a temporary string that may be written as the prefix string [prefix] followed by (or concatenated with) the base color (base color) of the current pixel: [prefix](base color).

There are several ways of identifying the set of candidate strings from the compression dictionary and the temporary string, [prefix](base color). In one way, a candidate string is any string in the compression dictionary that has a candidate prefix string [candidate prefix] followed by the true color index [TC_C], in which the candidate prefix string [candidate prefix] exactly matches the prefix string [prefix] in the temporary string, and in which the true color corresponding to the true color index [TC_C] approximately matches the base color (base color) of the current pixel. The true color may approximately match a base color if the true color is within a color space threshold τ_{cs} of the base color.

As discussed above, a prefix string may be a sequential ordered list of indices from the color table. Alternatively, the prefix string may be any sequential ordered list of codes from the compression dictionary. Thus, to have an exact match between the candidate prefix string and the prefix string, the two prefix strings must be of the same length and each code or index at a position in the candidate prefix string must correspond to the code or index at that same position in the prefix string. For example, the candidate prefix string [candidate prefix] = [34, 22, 56] exactly matches the prefix string [prefix] = [34, 22, 56]. In contrast, the candidate prefix string [candidate prefix] = [34, 22, 56] does not exactly match the prefix strings [prefix] = [35, 22, 56] or [34, 22].

The color space threshold τ_{cs} may be set before operation of the compression process 100. In another implementation, the predetermined color space threshold τ_{cs} may be preset before the current pixel's set of candidate strings is identified at step 116 to take into account neighboring pixel's candidate strings or to take into account the current pixel's index value. For example, if it is determined at step 114 that the true color nearest to the base color lies outside of the threshold τ_{cs}, then the threshold τ_{cs} may increased to provide for at least one candidate string at step 116.

In another way of identifying the set of candidate strings from the compression dictionary and the temporary string, a candidate string is any string in the compression dictionary that has a candidate prefix string [candidate prefix] followed by a true color code [TC_C], in which the candidate prefix string [candidate prefix] approximately matches the prefix string [prefix] in the temporary string, and in which the true color corresponding to the true color code [TC_C] approximately matches the base color (base color) of the current pixel.

As above, the true color may approximately match a base color if the true color is within a predetermined color space threshold τ_{cs} of the base color. However, in this identification alternative, the candidate prefix string [candidate prefix] may be selected to approximately match the prefix string [prefix] in the temporary string. Using the example above, the candidate prefix string [candidate prefix] = [34, 22, 56] approximately matches the prefix strings [prefix] = [35, 22, 56] or [34, 25, 55]. The process 100 may operate by determining whether each of the codes in the candidate prefix string [candidate prefix] lies within a threshold of each of the codes at the same position in the prefix string [prefix].

In another way of identifying the set of candidate strings, an aggregate difference (that may be normalized by length) between the temporary string and the candidate string may be tested.

If the set of candidate strings contains more than one string (step 118), then one of the candidate strings is selected (step 120). The one candidate string may be selected at step 120 according to the state of the compression dictionary. In particular, the candidate string is selected to be that string whose selection would cause a minimum increase in the size of the compression dictionary. The one candidate string may be selected at step 120 by examining all possibilities, performing a tree search through the possible strings, and picking a path that is longest, thus causing a minimum increase in the size of the compression dictionary.

Once the candidate string is selected (step 120), the prefix string is set to be the candidate string (step 122). In this way, a new string will be built from previously known data strings plus one new data value after the next pixel is selected as the current pixel (step 124), and the process returns to identifying a base color of the current pixel (step 112).

If the set of candidate strings does not contain more than one candidate string (step 118), and if the set of candidate strings contains a single candidate string (step 126), then the prefix string is set to be the candidate string (step 122), and the next pixel is selected as the current pixel (step 124).

If the set of candidate strings does not contain a single candidate string (step 126), then the process next considers as the candidate string a candidate prefix string plus the identified index value from step 114 (step 128). If the candidate string is in the compression dictionary (step 130), then the prefix string is set to be the candidate string (step 122) and the next pixel is selected as the current pixel (step 124).

If the candidate string is not in the compression dictionary (step 130), this indicates that a new pattern has been identified. Thus, an entry is added to the compression dictionary (step 132). That newly-assigned entry maps a new code to the candidate string. In this way, when the process sees that candidate string in the future, the process outputs its code.

Next, the code representing the prefix string is output to the compressed representation (step 134). As mentioned above, if, for the previous pixel, a candidate string was identified, then the prefix string is set to be the candidate string, and the code for the candidate string (assigned in step 132) is output to the compressed representation at step 134. Next, the prefix string is set to be the identified index value from step 114 (step 136). Thus, if, for the previous pixel, a candidate string was not identified, then the prefix string was set to be the identified index value (step 114), and the code indicating the true color from the color table that is nearest to the base color of the previous pixel is output to the compressed representation at step 134.

In any case, after the prefix string is set (step 136 or 122), the process selects the next pixel in the image as the current pixel (step 124).

The dithering process permits an improvement in image quality that may be reproduced during decompression. Moreover, the dithering process permits the use of a higher predetermined color space threshold τ_{cs}.

The compressed representation output from the process 100 may be decompressed using a decompression method corresponding to the process 100. For example, the compressed representation output from the process 100 may be decompressed using standard LZW decompression if the compressed form was created by the particular implementation just described.

The techniques, methods, and systems described here may find applicability in any computing or processing environment in which electronic data may be compressed. Various implementations of the systems and techniques described here may be realized in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations thereof.

A system or other apparatus that uses one or more of the techniques and methods described here may be implemented as a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer system to operate on input and/or generate output in a specific and predefined manner. Such a computer system may include one or more programmable processors that receive data and instructions from, and transmit data and instructions to, a data storage system, and suitable input and output devices. Each computer program may be implemented in a high-level procedural or object-oriented programming language, or in assembly or machine language if desired; and in any case, the language may be a compiled or interpreted language. Suitable processors include, by way of example, both general and special purpose microprocessors.

Generally, a processor will receive instructions and data from a read-only memory and/or a random access memory. Storage devices suitable for tangibly embodying computer instructions and data include all forms of non-volatile memory, including semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM disks.

These elements also can be found in a conventional desktop or workstation computer as well as other computers suitable for executing computer programs implementing the methods described here, which can be used in conjunction with any compression software, or any other software capable of compressing data. Any of the foregoing may be supplemented by, or implemented in, specially designed ASICs (application specific integrated circuits). The systems and techniques may be implemented as a standalone utility or plugin utility. Likewise, the systems and techniques may be implemented at the operating system level or as a feature of an application.

A number of embodiments have been described. Nevertheless, it will be understood that various modifications may be made without departing from the spirit and scope of the invention. For example, advantageous results still could be achieved if steps of the disclosed techniques were performed in a different order and/or if components in the disclosed systems were combined in a different manner and/or replaced or supplemented by other components. Accordingly, other embodiments are within the scope of the following claims.

For example, the base color of the selected pixel may be the true color of the selected pixel. In this case, there is no dithering of the data. The one candidate string may be selected

at step 120 by selecting the candidate string for which the true color best matches the base color.

The color table may be adapted to the particular image being compressed.

The predetermined color space threshold τ_{cs} may be set dynamically as pixels are considered during the process 100. For example, the local noise level near a current pixel may alter the predetermined color space threshold τ_{cs}. If the local noise level is high, then the predetermined color space threshold τ_{cs} may be increased.

The compressed data may be decompressed by a method according to any color table-based compression technique such as, for example, the deflate algorithm, which is commonly used in PNG formats. Basically, for any color table-based compression technique, the index values that are input into the technique are replaced by the true colors. Additionally, where the color table-based compression technique would normally compare an index value to elements in the compression dictionary to identify an exact match, the color table-based compression technique would now compare a true color (or base color) to elements in the compression dictionary to identify one or more candidates that approximately match the true color. Then, if there is more than one candidate, the technique is modified to select the candidate that produces the smallest compression dictionary. Moreover, the dithering technique detailed above with respect to the LZW compression technique may be applied to any color table-based compression technique.

What is claimed is: